ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE SERVICIOS DE RED CON HERRAMIENTAS DE DEVOPS

CREACIÓN DE UNA APLICACIÓN CON DOCKER MEDIANTE UN FRAMEWORK DE PHP

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN REDES Y TELECOMUNICACIONES

ESTEBAN GABRIEL LÓPEZ BASANTES

esteban.lopez@epn.edu.ec

DIRECTOR: ING. FERNANDO VINICIO BECERRA CAMACHO

fernando.becerrac@epn.edu.ec

DMQ, AGOSTO 2023

CERTIFICACIONES

Yo, Esteban Gabriel López Basantes declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Esteban Gabriel López Basantes

esteban.lopez@epn.edu.ec

loesteban2422@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por Esteban Gabriel López Basantes, bajo mi supervisión.

Fernando Vinicio Becerra Camacho
DIRECTOR

fernando.becerrac@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Esteban Gabriel López Basantes

DEDICATORIA

A mi primo Andrés Fernando, quien ha sido mi constante apoyo a lo largo de este camino, brindándome razones para seguir firme. Incluso en aquellos días en los que me sentía tan vulnerable y cada paso parecía un retroceso en lugar de un avance, tú estuviste ahí para brindarme ánimo. Hubo momentos en los que contemplé dejar esta vida, pero tu confianza me animó a seguir adelante, a reconocer lo que yo valía y a apreciar todo aquello por lo que debo estar agradecido.

Quiero expresarte, primo, mi más profundo agradecimiento. Tú fuiste la persona que me salvo la vida. Y hoy por hoy no puedo imaginar dónde estaría si no hubieras sido mi apoyo en cada ciclo de mi vida.

En un futuro próximo me propongo como objetivo el compartir una cena en Europa comiendo carnes, tomándonos un vino y riéndonos de cada persona que me decía que nunca lograría ser nada en la vida, tal como alguna vez soñamos. Gracias, primo, por todo lo que has hecho y significas para mí.

Esteban.

AGRADECIMIENTO

Esencialmente, quiero agradecerme a mí mismo, ya que solo yo sé lo que me costó estar en donde estoy. Tener días en los que quería dejar la carrera, los estudios, los objetivos que alguna vez me propuse en la vida, el levantarme, verme al espejo y decirme a mí mismo: "Vamos, Esteban, tú puedes ser mejor de lo que fuiste ayer".

El motivarme y volver a aprender todo por mi cuenta, ya que lo que la vida real me ha enseñado es que el esperar no transforma, el no tener objetivos nuevos no promueve un avance en nuestras vidas, y si queremos superarnos cada día, debemos hacerlo por nuestra cuenta, ya que nadie más lo hará por nosotros.

Al escribir estas palabras, me llena de gratitud saber que he sido capaz de superar mis propias expectativas y que he demostrado a mí mismo que soy capaz de alcanzar metas que alguna vez parecían inalcanzables.

Quizás existan personas que me juzguen por la forma en que me expreso en estos momentos, pero la realidad es que todo lo que tengo hoy por hoy me lo gané mediante lágrimas, tristezas y el sacrificar salidas a beber, por el tomar un libro y quedarme estudiando hasta altas horas de la madrugada, pero con la satisfacción de seguir creciendo intelectual y éticamente.

Por último, me gustaría comprometerme a seguir esforzándome y dando lo mejor de mí en cada nuevo desafío que la vida presente. Estoy seguro de que esta experiencia académica es solo el comienzo de una vida llena de éxitos y crecimiento personal.

Esteban.

ÍNDICE DE CONTENIDOS

RESUMEN

El presente proyecto de titulación se enfoca en la creación de una imagen base utilizando *Docker*, con el objetivo de facilitar el despliegue de una tienda virtual desarrollada con el *framework Laravel*. La imagen base está compuesta por PHP, MySQL y *Composer*; tecnologías que se detallan en una sección posterior

La primera sección incluye los objetivos generales y específicos que se han planteado para este proyecto de titulación. Así como también se detalla el alcance del proyecto y el marco teórico que sustenta su implementación.

La segunda sección describe la metodología aplicada para alcanzar los objetivos planteados. Se abordan los problemas identificados y se expone detalladamente la solución que ha sido desarrollada.

En la tercera sección de este documento, se describe los resultados obtenidos del despliegue de la aplicación. Se justifica el uso de tecnologías como: *Docker, Laravel, MySQL y Composer* así como la definición de la estructura de comandos para el funcionamiento del *Dockerfile*.

La cuarta y quinta sección exponen las conclusiones y recomendaciones que se obtuvieron a partir de la implementación del proyecto de titulación.

La sexta sección se presentan las referencias bibliográficas que respaldan la investigación y desarrollo del proyecto. Estas fuentes han sido fundamentales para sustentar teóricamente el trabajo y garantizar la validez de las propuestas presentadas.

Finalmente, los anexos que complementan el trabajo realizado se encuentran dentro de la séptima y última sección de este documento.

PALABRAS CLAVE: *Dockerfile, Framework, Laravel*, contenedores.

The present graduation project focuses on the creation of a foundational image using Docker, with the aim of facilitating the deployment of an e-commerce store developed with the Laravel framework. The foundational image is composed of PHP, MySQL, and Composer; technologies that are detailed in a subsequent section within this same document.

The first section provides a comprehensive description of what has been developed, including both the overarching and specific objectives set for this graduation project. Additionally, the project's scope and the theoretical framework underpinning its implementation are detailed.

The second section outlines the methodology applied to achieve the stated objectives. It addresses identified issues and thoroughly presents the solution that has been developed.

In the third section of this document, the obtained results from the application deployment are described. The rationale for utilizing technologies such as Docker, Laravel, MySQL, and Composer is justified, along with the definition of command structures for the functionality of the Dockerfile.

The fourth and fifth sections present the conclusions and recommendations derived from the implementation of the graduation project.

The sixth section showcases the bibliographical references that support the research and development of the project. These sources have been essential in theoretically substantiating the work and ensuring the validity of the proposed approaches.

Lastly, the appendices that complement the executed work are located within the seventh and final section of this document.

KEYWORDS: *Dockerfile, Framework, Laravel, containers.*

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

En el presente proyecto se presenta el despliegue de una aplicación web mediante *Docker*, identificando desafíos, problemas y áreas de mejora en términos de portabilidad y escalabilidad. A partir de este análisis, se diseña y desarrolla un enfoque que permite un ambiente de despliegue seguro y controlado.

Mediante la creación de imágenes *Docker* se encapsula el *stack* tecnológico y las dependencias específicas de *Laravel*. Esto permite la replicación similar del entorno en distintas fases del ciclo de vida de la aplicación y en distintos entornos de implementación.

Los resultados y logros del proyecto aportan conocimiento valioso a la comunidad de desarrollo, destacando así la importancia y el impacto del encapsulamiento de aplicaciones web.

1.1 Objetivo general

Desplegar una aplicación *web* desarrollada en el *framework Laravel* utilizando *Docker* como herramienta de administración de contenedores.

1.2 Objetivos específicos

- Analizar la herramienta de *DevOps* que soluciona cada componente del proyecto de titulación.
- Diseñar la solución para cada servicio de *networking* mediante herramientas de *DevOps*.
- Implementar las soluciones mediante herramientas *DevOps* para el despliegue de los servicios de *networking*.
- Verificar el funcionamiento de cada servicio de *networking* implementado mediante *DevOps*.

1.3 Alcance

El presente proyecto pretende que los estudiantes utilicen DevOps y desplieguen las soluciones en host o en equipos de *networking*. Para realizar este propósito se necesita de herramientas,*DevOps* capaces de automatizar el despliegue de los servicios propuestos y además se puedan realizar pruebas de las soluciones. En el proyecto se pretende crear una aplicación con Docker mediante un framework de PHP.

1.4 Marco Teórico

Development & Operation (DevOps)

DevOps es una cultura de trabajo que involucra personas, procesos y herramientas cuyo eje es la integración y colaboración de los equipos de desarrollo (Dev) con los equipos de infraestructura (Ops) [1].

Para qué utilizar *DevOPS*

DevOps tiene como objetivo principal es desplegar *software* de calidad en el menor tiempo posible permitiendo a las compañías tener equipos de TI de alto rendimiento [1].

Se presenta varios tipos de herramientas de manejo entre los que destacan:

- **Cultura colaborativa**: El primer paso para la adaptación *DevOps* es fomentar una cultura en la cual se pueda compartir, colaborar y realizar trabajos en equipo entre los desarrolladores, operadores y cualquier persona que sea fundamental dentro del sistema [1].
- **Automatización:** Es un proceso clave en *DevOps* que reduce el factor de error humano, mediante el uso de herramientas de *software* que se encarga de realizar tareas repetitivas, garantizando mayor confiabilidad del sistema y optimizando el tiempo que demanda el despliegue de una aplicación [1].
- **Implementación en la nube:** Adoptar este tipo de servicios brinda escalabilidad y flexibilidad para despliegues rápidos y eficientes [1].
- **Mejora continua**: Es una característica fundamental en *DevOps* que permite analizar, optimizar, corregir y reestructurar la aplicación cuando esta así lo requiera [1].

Docker

Es una tecnología que simplifica la implementación y administración de aplicaciones al encapsular el código, dependencias y librerías en contenedores [2].

Docker plantea un modelo basado en imágenes que facilitan la implementación en varios entornos. Esta tecnología permite distribuir los servicios y dependencias de la aplicación sin comprometer su funcionamiento. Además, se encarga de automatizar aplicaciones o conjuntos de procesos que constituyen un entorno de contenedores [2].

Contenedores

El contenedor es un apoyo para los desarrolladores dado que facilita la creación, pruebas e implementación de aplicaciones antes de ser lanzadas al mercado. El principal beneficio radica en la facilidad y portabilidad de la implementación de las aplicaciones [2] [3]. Los contenedores son altamente portátiles, ya que pueden ser ejecutados en diversas arquitecturas de manera independiente. Este estilo es implementado como norma en varias empresas, ya que la utilización de un contenedor es capaz de garantizar el uso compartido de recursos [3].

MySQL

Es un sistema que brinda soporte para el control de las bases de datos, al ser multiplataforma facilita la implementación en la mayoría de los sistemas operativos por lo cual es eficiente en el encapsulamiento que se ha requerido para el despliegue de la aplicación [3].

Composer

Es un gestor de dependencias de PHP utilizado para la instalación de *Laravel*, facilita la incorporación, administración y actualización de paquetes de terceros que se generen dentro del proyecto [4].

Laravel

Es un F*ramework* de PHP que permite que la programación en este lenguaje sea elegante y simple ya que tiene una estructura de directorios definida lo que permite modular la aplicación [5].

2 METODOLOGÍA

El proyecto se basa en el cumplimiento de cinco ejes fundamentales, los cuales convergen para establecer una arquitectura eficiente destinada al despliegue de la aplicación *web*. La metodología empleada se presenta en la Figura 2.1, cuyo diseño ha sido estudiado para asegurar el cumplimiento de los objetivos planteados en la implementación de la aplicación.

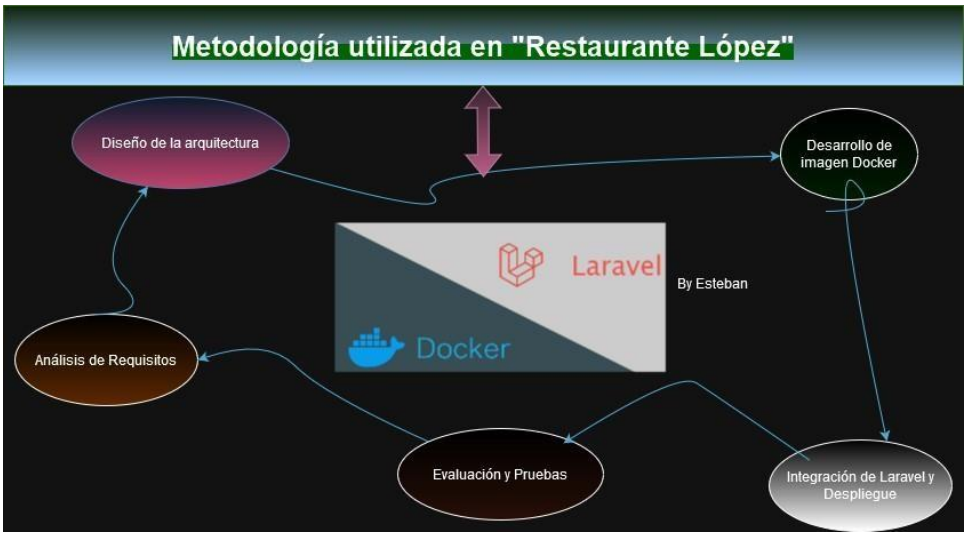


Figura 2.1 Metodología utilizada

Para el análisis de requisitos se realizó una revisión literaria de cada uno de los conceptos claves relacionados con *Docker*, *Laravel* y las formas en que se puede desplegar la aplicación contribuyendo así una base sólida de conocimientos sobre los componentes claves del proyecto.

Para el diseño de la arquitectura se llevó a cabo la planificación del entorno *Docker* y la infraestructura necesaria para el despliegue de aplicación *Laravel*. Esto implicó la identificación de componentes, servicios, contenedores y dependencias que son necesarias para lograr una implementación eficiente y escalable.

En el desarrollo de imagen se creó un archivo de instrucciones personalizadas para cada uno de los componentes de *stack* tecnológicos, incluyendo el servidor *web*, base de datos, dependencias y el entorno PHP para la creación de la imagen. Se configuró el archivo *Dockerfile* y se aplicaron prácticas de optimización y seguridad para garantizar un entorno de encapsulamiento eficiente y seguro.

Se realizaron pruebas para garantizar la funcionalidad en el entorno local y el de producción estas permitieron verificar el rendimiento, seguridad y funcionalidad que tenía la aplicación.

3 RESULTADOS

A continuación, se presentan los resultados obtenidos conforme a cada uno de los objetivos planteados.

3.1 Analizar las herramientas de *DevOps* que soluciona cada componente del proyecto de titulación

Como primer punto se realizará una investigación sobre el manejo de *DevOps*, contenedores, *Docker* y F*rameworks* ya que son herramientas con las cuales se procede a realizar el proyecto de titulación.

Introducción a *Docker*

Antes de la implementación de *Docker*, la forma en que se desarrollaba las aplicaciones involucraba máquinas virtuales, bibliotecas y problemas de funcionamiento debido a la no compatibilidad de algunas tecnologías ya que cada herramienta tenía una forma distinta de ser configurada. Es por lo que desplegar una aplicación es complicada, tardía y en algunos casos inoperante debido al poco entendimiento de las tecnologías entre sí [2].

Docker es una herramienta de código abierto que se encuentra diseñada, para mejorar el desarrollo, implementación y ejecución de aplicaciones. Por lo tanto, son considerados como entornos de software ligeros y portátiles en donde la función principal es la de encapsular todo el entorno de desarrollo para garantizar que puedan ejecutarse bajo un único idioma de configuración eliminando la no compatibilidad de las aplicaciones [2].

La implementación de *Docker* trae consigo varios conceptos clave que garantizan un correcto desarrollo en la gestión de aplicaciones como se muestra a continuación:

- **Contenedor:** Instancias aisladas y ejecutables de aplicación, son más livianos que montar máquinas virtuales, lo que integra un arranque rápido y un menor consumo de recursos [3].
- **Imagen:** Es un archivo compuesto de distintas capas que se utiliza en la ejecución del código dentro del contenedor de *Docker* [6].
- ***DockerFile*:** Es un recurso de configuración para crear distintas imágenes personalizadas con aplicaciones propias. Este archivo incorpora como mínimo un sistema operativo base necesario para el funcionamiento [2] [7].
- ***Docker Hub:*** Es un repositorio en donde los usuarios pueden almacenar, compartir y descargar imágenes [7].

- ***Docker compose*:** Es una herramienta dedicada a la orquestación, es útil para el despliegue y ejecución de aplicaciones en varios contenedores de forma fácil y rápida [2].

Finalmente, se puede considerar que *Docker Engine* es una tecnología que se encarga de vincular los contenedores con cada uno de los sistemas operativos permitiendo así construir, configurar y ejecutar cualquier contenedor dentro de un sistema operativo. Además, esta tecnología *Engine* proporciona la construcción de imágenes en base a la configuración de los archivos llamados *Dockerfile* [8].

En la Tabla 3.1, se presentan los principales componentes que permiten la construcción del archivo de instrucciones *Dockerfile*.

Tabla 3.1 Sintaxis *DockerFile* [7]

Comando	Que permite realizar	Ejemplo
FROM	Se emplea para definir la imagen de partida que formará la base de la imagen que se esta creando.	*FROM Ubuntu:22.04*
Run	Es un comando que me permite instalar dependencias	*RUN: npm install -g*
Entrypoint	Es utilizado para ejecutar un archivo *bash* que contenga utilidades de la aplicación	*ENTRYPOINT "start-container*
Workdir	Coloca el directorio de trabajo en el que se trabaja	*Workdir /var/www/html*
Copy	Copia la información de un archivo hacia un directorio especificado	*Copy start-container /var/www/html*
Expose	Permite la salida de la imagen en el puerto especificado	*Expose 8000*
Curl	Permite transferir datos entre un servidor y un cliente	*Curl https//composer.com*

Otras de las tecnologías dedicadas a la administración de contendores es *Podman*. Sus principales características y funcionalidades se detallan a continuación:

Podman

Podman es una herramienta *Open Source* administradora de contendores que permite la gestión y ejecución de contendores en Linux [9].

A diferencia de algunas soluciones similares, *Podman* ofrece un enfoque sin *daemon* lo que significa que no requiere un proceso en segundo plano para trabajar como lo realiza *Docker* [9]. Esto brinda ventajas en términos de seguridad y control dado que si cada contenedor que se encuentre ejecutando es un proceso independiente este no necesitara privilegios especiales para su ejecución [9].

Pods

Los *pods* se encargan del agrupamiento de contenedores para una ejecución conjunta con el compartimento de sus recursos. Su administración es realizada mediante CLI y una biblioteca llamada *libpod* lo cual proporciona las API para gestionar los contenedores, imágenes, *pods* y volúmenes [9].

Un *pod* se compone de una estructura única responsable de aislar los contenedores del *host*, cada contenedor único se encarga de rastrear procesos y supervisar aquellos que no estén funcionando correctamente asegurando que no se pueda eliminar si aún hay recursos en uso [9].

Podman es una solución actual que ha revolucionado contenedores pasados ya que ofrece capacidades de alto rendimiento, pero con flexibilidad, accesibilidad y seguridad que son altamente valoradas por los equipos de desarrollo TI por lo cual se presentan las siguientes ventajas [9]:

- **Gestión completa de imágenes:** *Podman* facilita la administración completa del ciclo de vida de las imágenes de contendores incluyendo la ejecución, conexiones de red, control y eliminación [9].
- **Ejecución segura sin privilegios:** Al no tener un *daemon* en segundo plano tiene la habilidad de ejecutar y aislar recursos para *pods* y contenedores sin requerir privilegios de super usuario permitiendo que inclusive los contenedores sean livianos [9].
- **Compatibilidad con imágenes *Docker*:** *Podman* admite tanto imágenes *Docker* como las que cumplen con los estándares de OCI, ofreciendo una interfaz de línea de comandos compatible con *Docker* [9].
- ***Api Rest*:** Permite acceder a características avanzadas de *Podman*, lo que facilita a una integración de flujos de trabajo complejos [9].

A través de la Tabla 3.2 se presentan algunas diferencias entre dos de los contenedores más utilizados para el despliegue de aplicaciones *web*:

Tabla 3.2 Comparación *Docker* vs *Podman*

Docker	***Podman***
Gestión de contenedores	No requiere privilegios de super usuario
Se implemento en la configuración del *daemon.*	Arquitectura libre de *daemons*
Docker abarca de manera integral la creación y administración de contendores	*Podman* junto con herramientas afines como *Buildah* y *Skopeo* se especializa en aspectos específicos de los contendores.

A pesar de estas diferencias, *Podman* es una alternativa eficaz junto con *Docker* para el encapsulamiento de aplicaciones. Además, ambas herramientas pueden coexistir de manera complementaria.

Finalmente, el escoger entre cualquiera de las dos herramientas dependerá de lo que se requiera crear. Para el caso del proyecto de titulación se utiliza *Docker* ya que es una tecnología conocida, por lo tanto, se presenta una mayor documentación y compatibilidad con una gran cantidad de tecnologías que permiten el despliegue de contenedores.

Frameworks

La palabra *Framework* o marco de trabajo se define como la agrupación de reglas, convenciones y estructuras que se encuentran destinados para el desarrollo de aplicaciones como [10], por ejemplo:

- Desarrollo de aplicaciones *web*.
- Creación de videojuegos.
- Simulación de entornos virtuales.
- Desarrollo de aplicaciones médicas.

Un *Framework* de desarrollo de aplicaciones *web* es el conjunto de bibliotecas, herramientas y estructuras que proveen la creación de aplicaciones y sitios *web* [10].

Por otro lado, los *Frameworks* presentan características de organización, estructuración de los códigos base lo que permite tener la parte colaborativa entre los desarrolladores

y la reutilización de los códigos originando un trabajo comunitario en el que no presenta reinvención constante de lo que ya se encuentra creado [10].

Frameworks en PHP

Cada *Framework* para el desarrollador web es una herramienta única que presenta características que permitan crear aplicaciones en base a las necesidades del consumidor para ello se considera los siguientes [10]:

- ***Laravel:*** Es particularmente conocido por la sintaxis que es de fácil entendimiento aún sin conocer extensamente sobre programación, ya que posee una gran cantidad de elementos básicos como: autenticación de los usuarios, administración de sesiones y almacenamiento en cache proporcionando una aplicación moderna y fácil de construir [5].
- ***CodeIgniter:*** Se utiliza para enfoques de aplicaciones altamente escalables, pero con un tamaño reducido ya que crea aplicaciones ligeras y sencillas para comenzar en PHP [10].
- ***Symfony:*** Fue lanzado por primera vez en 2005 por la empresa *SensioLabs* en el que se basó la construcción de *Yahoo Bookmarks* debido a que posee una licencia MIT que presenta un tipo de analogía con el *software* libre permisivo y se puede utilizar dentro de un *software* propietario [11].
- ***CakePHP:*** Representa un marco de desarrollo rápido para el lenguaje de programación PHP. Este *Framework* permite tener una estructura base que capacita a los desarrolladores para la creación de aplicaciones web, el objetivo de *Cake* es facilitar un entorno donde sea posible trabajar con organización y velocidad [12].

Una vez definidos algunos ejemplos de *Frameworks* en PHP y en conjunto con la documentación de cada uno de ellos se escoge *Laravel* por las razones que se describen a continuación:

Es un *Framework* de desarrollo web de código abierto con una sintaxis expresiva y elegante siendo en la actualidad de los más populares y utilizados en PHP [5]. La implementación de *Laravel* permite desarrollar un sistema web haciendo uso de la arquitectura MVC permitiendo tener el código ordenado, estructurado e integrado con las funciones de la aplicación [5].

Laravel incluye herramientas para el desarrollo de aplicaciones *web* entre las cuales se presentan las siguientes:

- **Sistema de rutas:** Ofrece un sistema de rutas intuitivo y fácil de usar que permite a los desarrolladores definir una ruta de manera simple [5].
- **Motor de plantilla:** *Laravel* ofrece la creación de plantillas reutilizables y flexibles para la interfaz del usuario con el denominado *"Blade"* [5].
- ***Eloquent* ORM:** Es conocido por sus siglas de *Object Relational Mapping* en el cual se permite interactuar con las bases de datos de manera sencilla, elegante y sin tanta complejidad como otros *Frameworks* [5] [13].
- **Migraciones de las bases de datos:** Es un sistema de administración de datos que proporciona una forma de crear, modificar o eliminar tablas de manera controlada [5].
- **Seguridad:** Incluye herramientas de seguridad integrada como la protección contra ataques de inyección SQL, CSRF, cifrados de contraseñas, encriptación y desencriptación de datos que permite tener confiabilidad en la aplicación [5].
- ***Testing*:** *Laravel* incluye herramientas para realizar pruebas automatizadas, lo que contribuye a un control en la calidad de código lo cual es fundamental para prevenir errores [5].
- **Comunidad Activa:** Lo importante de este *Framework* es la gran cantidad de recursos, desarrolladores, tutoriales y documentación que permite el crecimiento futuro de las aplicaciones [5].

3.2 Diseñar la solución para cada servicio de *networking* mediante herramientas *DevOps*

El presente trabajo de titulación busca crear un contenedor, en el cual se permita el despliegue de la aplicación web, por lo que como primer escenario se considerara la arquitectura que se plantea para formar el contenedor.

Una vez que se tiene un panorama adecuado de *Docker*, se procede a identificar las herramientas necesarias para implementar la aplicación dentro de un contenedor. Por medio del archivo *Dockerfile* se integra los argumentos para la construcción de la imagen base. Esta imagen permite el despliegue de servicios como MySQL, *Composer* y npm los cuales permiten construir todo el entorno de desarrollo de la aplicación.

Toda esta arquitectura se encuentra basada en un sistema operativo Linux con la imagen base de PHP 8.1. Esta imagen permite instalar los componentes y dependencias

necesarias en el archivo *Dockerfile* para el despliegue del contenedor. En la Figura 3.1 se presenta la arquitectura base del proyecto de titulación.

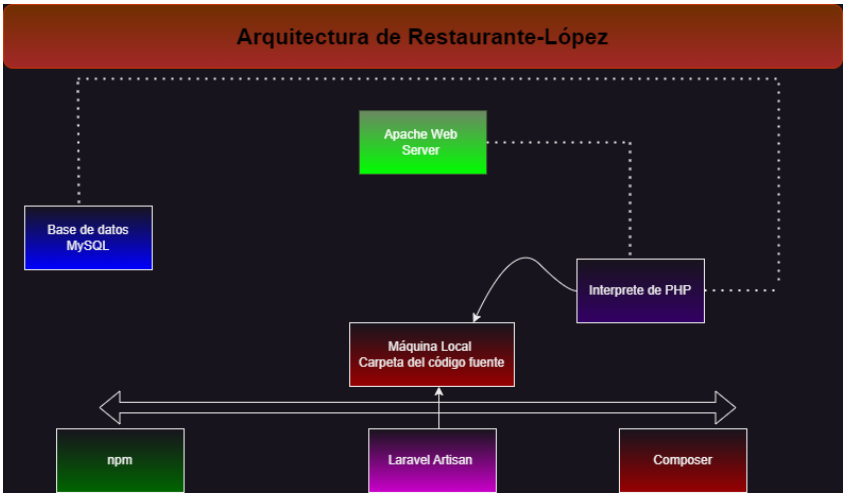


Figura 3.1 Arquitectura de la aplicación

Para finalizar, se ejecuta el contenedor de *Docker* con el propósito de comprobar el correcto funcionamiento de la aplicación de Laravel.

DockerFile

Es un archivo de texto que contiene las instrucciones para construir una imagen de *Docker* en el cual se especificaran las dependencias necesarias para la aplicación como es la versión de PHP, servidor web, y cualquier otra extensión que el proyecto requiera es por eso que es indispensable construir un archivo de texto adecuado a lo que se requiere como proyecto [14].

En la Figura 3.2 se puede apreciar un *Dockerfile* el cual ha utilizado como base la imagen de PHP con el servidor de apache, para establecer un entorno de desarrollo destinada a una aplicación web.

En cuanto a la línea número cuatro de la Figura 3.2 la etiqueta de contenedor es una instrucción esencial en un *Dockerfile*, ya que posibilita el mantenimiento y la administración de la imagen del contenedor. Aunque no es estrictamente necesario son consideradas como buenas prácticas *DevOps* proporcionar esta información.

```
1  #Se usa la imagen base de PHP con Apache
2  FROM php:8.1-apache
3  #Se coloca una etiqueta para identificar quien creó la imagen
4  LABEL maintainer="Esteban López"
```

Figura 3.2 Imagen base

Como siguiente escenario en la Figura 3.3, se actualiza el repositorio con *apt-get* ya que es recomendable siempre que se inicie con una distribución actualizar el sistema ya que de no ser así no se podrá continuar con la construcción del *Dockerfile.* Seguidamente de lo descrito anteriormente se instalan varias dependencias y herramientas como las bibliotecas de libpng y libjpeg que permiten la construcción de la imagen, finalmente se configura y se instala extensiones de PHP como *pdo* y *pdo_mysql* las cuales son fundamentales en cualquier proyecto para la conexión con la base de datos.

```
6   # Siempre se actualiza el sistema e instala dependencias
7   RUN apt-get update && apt-get install -y \
8       libpng-dev \
9       libjpeg-dev \
10      zip \
11      unzip \
12      git \
13      && docker-php-ext-configure gd --with-jpeg \
14      && docker-php-ext-install gd pdo pdo_mysql
```

Figura 3.3 Se actualiza el sistema e instala algunas dependencias

Se establece el directorio de trabajo en el *path* ***var/www/html*** dentro del contenedor como se muestra en la Figura 3.4, es allí donde se establecerán todas las líneas de código por parte de *Laravel*.

```
9   #Establecemos el directorio de trabajo dentro del contendor
10  WORKDIR /var/www/html
```

Figura 3.4 Directorio de trabajo.

En la línea número veinte como se presenta en la Figura 3.5, copia todos los archivos que se encuentran creados dentro del proyecto de *Laravel para* que puedan ser exportados junto con el contenedor.

```
19  # Copia los archivos del proyecto
20  COPY . .
```

Figura 3.5 Descarga dependencias PHP

En la Figura 3.6 se procede con la descarga de *composer*, una herramienta importante para gestionar las dependencias de *Laravel*, esta a su vez permite crear el proyecto dentro del contenedor.

Con respecto al pipe se lo utiliza para redirigir una salida del comando anterior hacia la entrada de otra línea de código lo que me permite tener un esquema más ordenado, en este caso se instala *composer* en el directorio /usr/local/bin el cual se encuentra dentro del contenedor.

```
22  # Se instala las dependencias de Composer
23  RUN curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --filename=composer
```

Figura 3.6 Descarga e instalación de composer

Se ejecuta el comando presentado en la Figura 3.7 para instalar cada una de las dependencias del proyecto utilizando la herramienta de *composer*.

```
25  # Se instala composer
26  RUN composer install
```

Figura 3.7 Instalación de algunas dependencias

En la Figura 3.8 se copia el archivo de configuración del servidor apache el cual se nombra como apache-config.conf en donde se crea algunas reglas que me permitan la conexión con el servidor.

Por otro lado, la instrucción número treinta y dos se encarga de habilitar el módulo de reescritura de la URL en el servidor web de apache que se encuentra configurada dentro del contenedor. Sin esta instrucción la aplicación web que se encuentra encapsulada no permitiría el uso de capacidades como la reescritura de URL proporcionadas por el servidor.

```
28  # Copia el archivo de configuración del servidor Apache
29  COPY apache-config.conf /etc/apache2/sites-available/000-default.conf
30
31  # Habilita el módulo de reescritura de Apache
32  RUN a2enmod rewrite
```

Figura 3.8 Copia del archivo de configuración de apache y reescritura dentro del servidor

La aplicación se puede conectar mediante el puerto 8000 como se muestra en la Figura 3.9, respecto al CMD permite que el *script* se ejecute cuando el contendor se inicie. Esto es lo que mantendrá al contenedor en funcionamiento mientras el servidor web se encuentre activo.

Por último, es crucial comprender que al construir un *Dockerfile*, el orden de escritura desempeña un papel fundamental. Si la estructura del *Dockerfile* no se encuentra bien organizada al momento de compilarlo, esto podría resultar en la imposibilidad de crear la imagen base.

```
34  # Se expone la aplicación en el puerto 8000
35  EXPOSE 8000
36
37  # Aqui se puede iniciar el servidor Apache como otras variabes
38  CMD ["apache2-foreground"]
```

Figura 3.9 Puerto utilizado

3.3 Implementar las soluciones mediante herramientas *DevOps* para el despliegue de los servicios de *networking*

Con la ayuda de *Docker* instalada de manera local se construye la imagen del proyecto con el comando ***Docker build -t docker-esteban-php .*** como se muestra en la Figura 3.10.

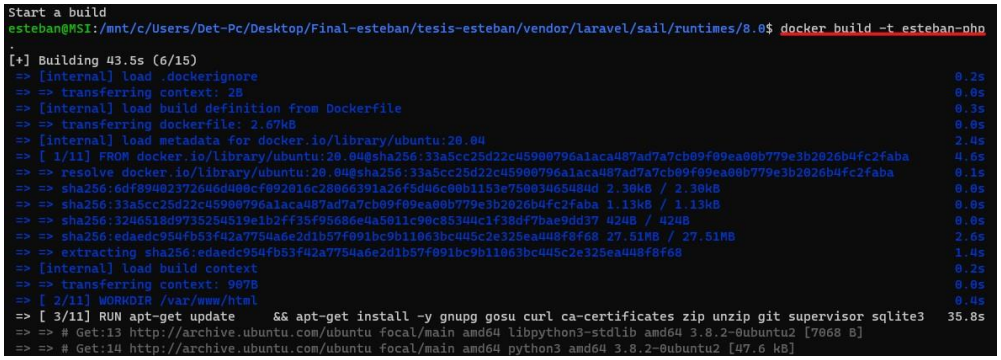


Figura 3.10 Construcción de la imagen base

Tal como se observa en la Figura 3.11, por medio del comando ***Docker images*** se verifica que la imagen este correctamente creada.

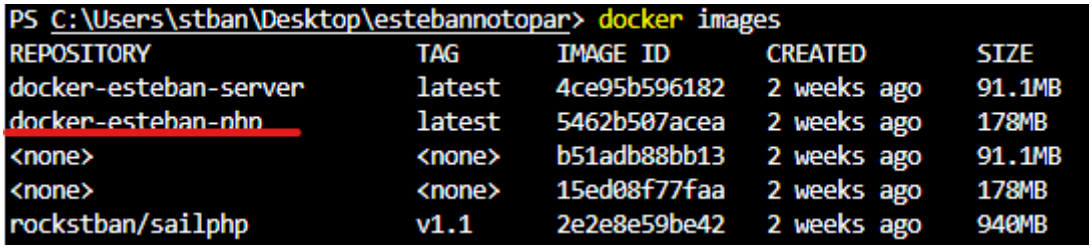


Figura 3.11 Se tiene imágenes que se han ido creando para levantar el proyecto

Si la imagen se encuentra funcionando se tendrá el despliegue del proyecto el cual se puede verificar con cada uno de los contenedores que se encuentran en funcionamiento gracias a la imagen base que se creó como se muestra en la Figura 3.12.

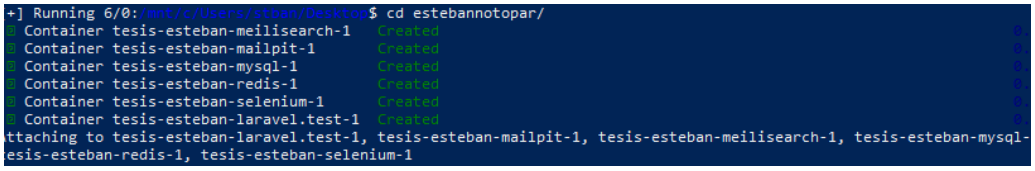


Figura 3.12 Se muestra el despliegue de los contenedores que fueron creados para la aplicación.

3.4 Verificar el funcionamiento de cada servicio de *networking* implementado mediante *DevOps*

Existen dos formas de iniciar el proyecto la primera es mediante el comando resaltado en rojo en la Figura 3.13, la cual inicia el servidor en el puerto 8000 como se mostróen la Figura 3.9.

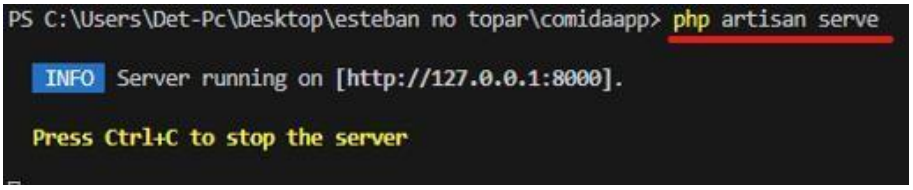


Figura 3.13 Levantamiento servicio forma uno

La otra forma es utilizar una distribución de *Linux* dentro de *Windows* para esto se utiliza WSL tal y como se muestra en la Figura 3.14, para ingresar a la distribución de *Ubuntu*.

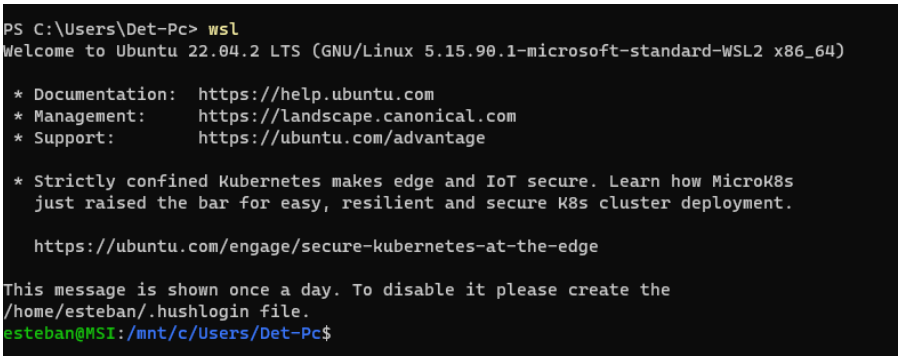


Figura 3.14 Utilizando la distribución de Ubuntu.

En la Figura 3.15 se establece la ruta en la cual se encuentra el proyecto mediante el comando ***"cd ruta"*** para construir la aplicación *web*.

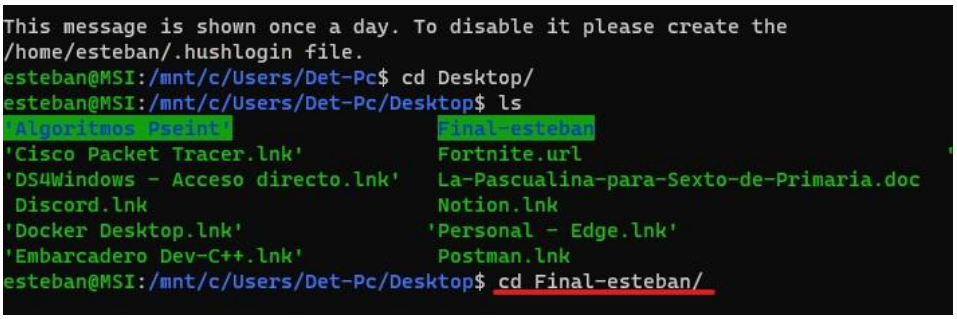


Figura 3.15 Ingreso al directorio para copiar el proyecto

Una vez en la ruta establecida en la Figura 3.15 se utiliza un *sail up* para el levantamiento de los contenedores tal y como se muestra en la Figura 3.16, es válido mencionar que *sail* está configurado como alias para levantar el proyecto de una manera rápida.

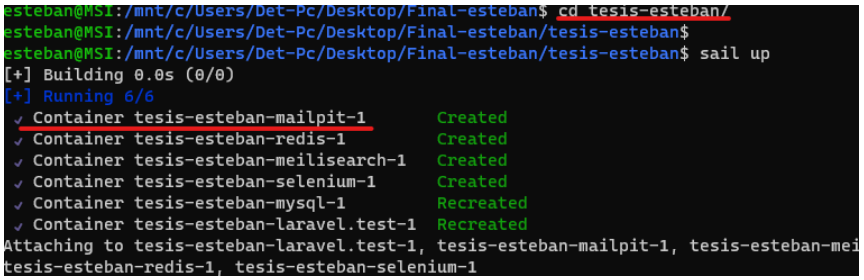


Figura 3.16 Levantamiento del proyecto

Para poder colocar el alias de manera permanente se ingresa en el archivo de configuración de bash tal y como se muestra en la Figura 3.17.

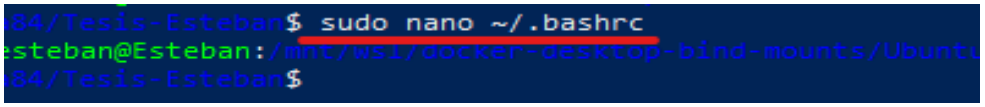


Figura 3.17 Modificación del archivo bash para poder crear un alias

Al ingresar en el archivo de configuración se escribe una línea en la que se coloque el alias para levantar el proyecto sin tener que escribir php artisan serve en este caso el alias es "*sail*" como se muestra en la Figura 3.18, a continuación, se guarda la configuración y se actualiza el archivo como se indica en la Figura 3.19.

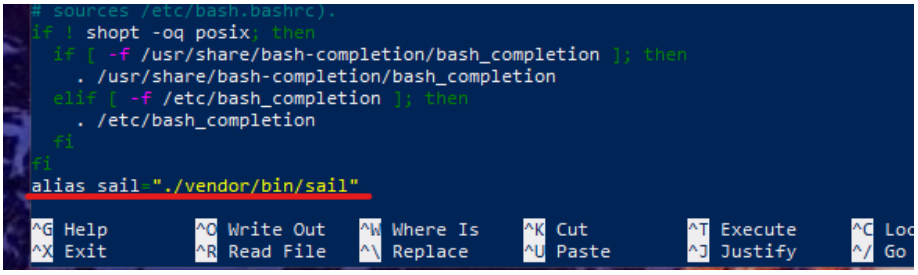


Figura 3.18 Creación de alias para levantar el proyecto de una forma rápida

Figura 3.19 Se guarda la configuración y se reinicia el proyecto Laravel

Para poder verificar el contenedor que se encuentra funcionando correctamente se utiliza el comando ***"Docker ps –a"*** el cual muestra todos los contenedores que han sidolevantados como se puede apreciar en la Figura 3.20, se encuentra el *entrypoint o cmd*y el puerto de salida que se empleó en la Figura 3.9 anteriormente descrito en el proyecto.

Figura 3.20 Contenedor principal en funcionamiento

Mediante el comando ***"Docker images"*** se tiene la imagen base que fue creada para realizar el proyecto como se muestra en la Figura 3.21.

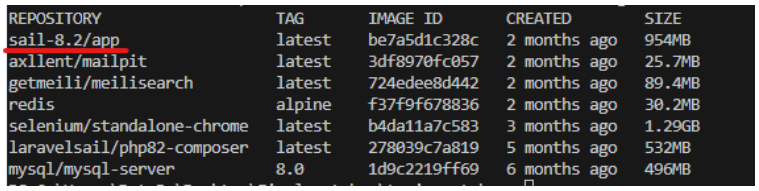


Figura 3.21 Verificación de la imagen creada

Para crear la base de datos con los usuarios, contraseñas, productos y cada unade las tablas implementadas en el proyecto se utiliza el comando resaltado en rojo en laFigura 3.22.

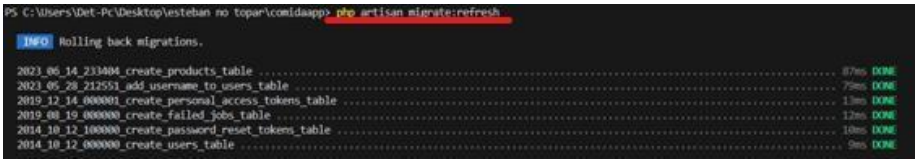
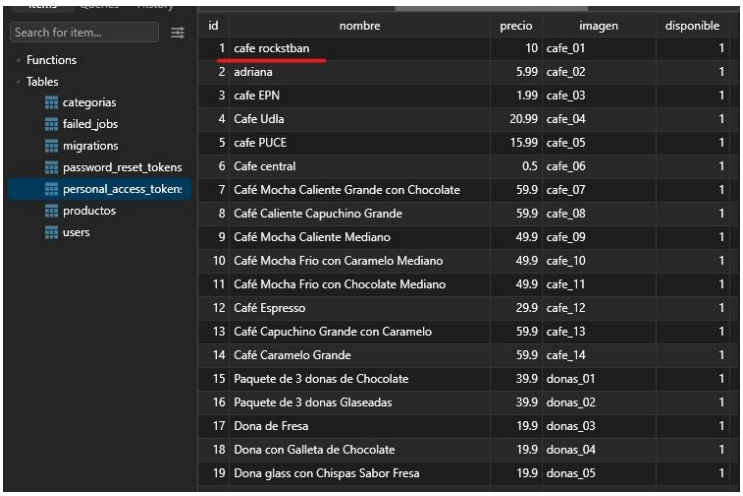


Figura 3.22 Creación de tablas de usuarios

Para la verificación de los datos se utiliza la herramienta Tableplus como se muestra en la Figura 3.23 en donde se considera todas las tablas que se creó con sus respectivas migraciones.



id	nombre	precio	imagen	disponible
1	cafe rockstban	10	cafe_01	1
2	adriana	5.99	cafe_02	1
3	cafe EPN	1.99	cafe_03	1
4	Cafe Udla	20.99	cafe_04	1
5	cafe PUCE	15.99	cafe_05	1
6	Cafe central	0.5	cafe_06	1
7	Café Mocha Caliente Grande con Chocolate	59.9	cafe_07	1
8	Café Caliente Capuchino Grande	59.9	cafe_08	1
9	Café Mocha Caliente Mediano	49.9	cafe_09	1
10	Café Mocha Frio con Caramelo Mediano	49.9	cafe_10	1
11	Café Mocha Frio con Chocolate Mediano	49.9	cafe_11	1
12	Café Espresso	29.9	cafe_12	1
13	Café Capuchino Grande con Caramelo	59.9	cafe_13	1
14	Café Caramelo Grande	59.9	cafe_14	1
15	Paquete de 3 donas de Chocolate	39.9	donas_01	1
16	Paquete de 3 donas Glaseadas	39.9	donas_02	1
17	Dona de Fresa	19.9	donas_03	1
18	Dona con Galleta de Chocolate	19.9	donas_04	1
19	Dona glass con Chispas Sabor Fresa	19.9	donas_05	1

Figura 3.23 Tabla de productos que fue desplegada en el proyecto

Se puede visualizar los usuarios registrados en la base de datos de la aplicación mediante aplicaciones como MySQL como se muestra en la Figura 3.24.

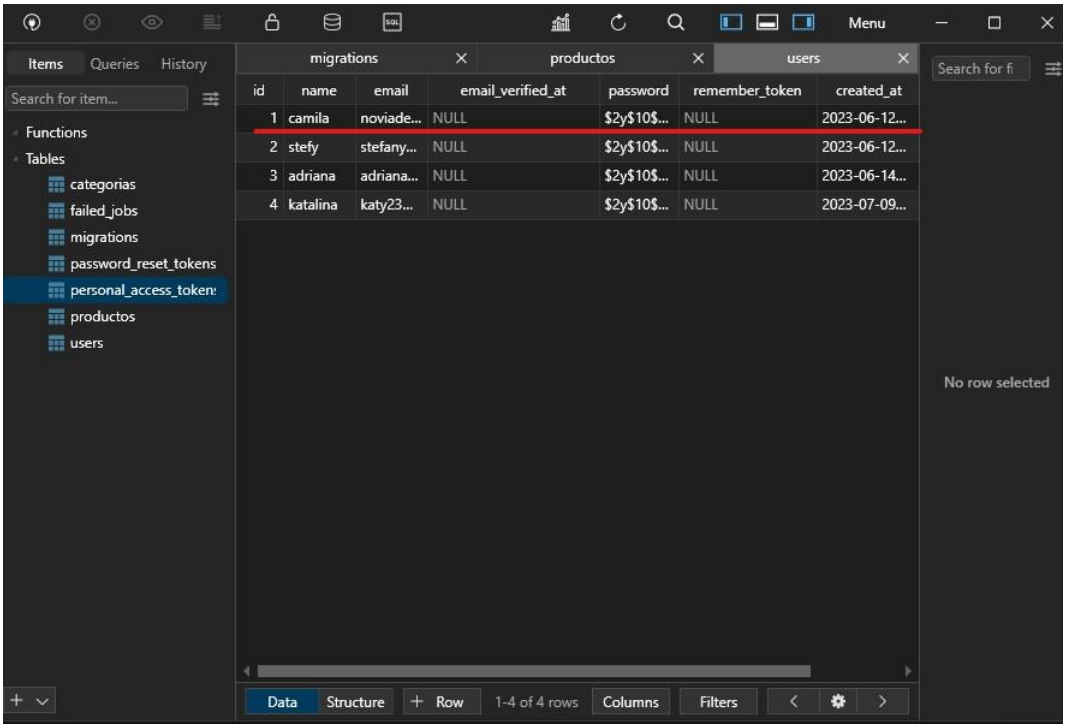


Figura 3.24 Tabla de usuarios

El archivo de configuración .env es muy importante ya que a través de él se controla toda la gestión de la información en MySQL a su vez del puerto en donde escucha la base de datos con su respectivo usuario y contraseña como se muestra en la Figura 3.25.

```
7   LOG_CHANNEL=stack
8   LOG_DEPRECATIONS_CHANNEL=null
9   LOG_LEVEL=debug
10
11  DB_CONNECTION=mysql
12  DB_HOST=mysql
13  DB_PORT=3306
14  DB_DATABASE=tesis_esteban
15  DB_USERNAME=sail
16  DB_PASSWORD=password
17  FORWARD_DB_PORT=3307
18
```

Figura 3.25 Archivo *env* que contiene la clave de la base de datos

Para comprobar que el *Dockerfile* funciona tanto de manera local como en producción se utiliza la herramienta *git* para clonar el proyecto y que pueda ser implementado en otro computador que no tenga instalada ninguna aplicación como se muestra en la Figura 3.26.

```
PS C:\WINDOWS\system32> choco install git
Chocolatey v1.4.0
Installing the following packages:
git
By installing, you accept licenses for the packages.
Progress: Downloading git.install 2.41.0... 100%
Progress: Downloading git 2.41.0... 100%

git.install v2.41.0 [Approved]
git.install package files install completed. Performing other installation steps.
The package git.install wants to run 'chocolateyInstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint):
```

Figura 3.26 Instalación de *Git*

Se comprueba que no se tiene instalado ningún archivo en la computadora Figura 3.27, para poder colocar una nueva ubicación de directorio.

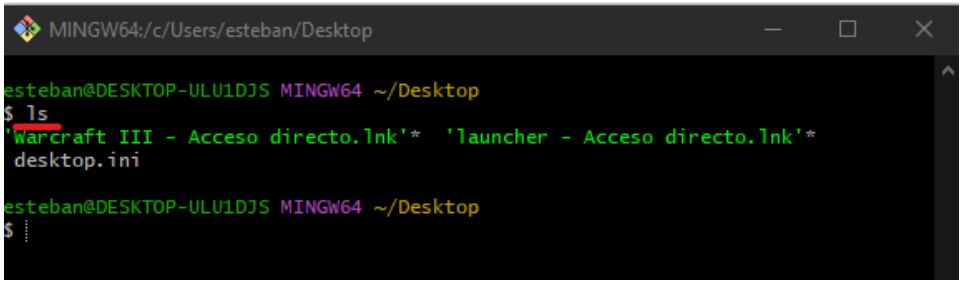


Figura 3.27 Ubicación en el directorio

Se crea una carpeta en donde se guardará el proyecto con el comando ***"mkdir"*** y se enlista ***"ls"*** como se tiene en la Figura 3.28.

```
esteban@DESKTOP-ULU1DJS MINGW64 ~/Desktop
$ ls
'Warcraft III - Acceso directo.lnk'*  'launcher - Acceso directo.lnk'*
 desktop.ini

esteban@DESKTOP-ULU1DJS MINGW64 ~/Desktop
$ mkdir compu2-tesis

esteban@DESKTOP-ULU1DJS MINGW64 ~/Desktop
$ ls -alh
total 21K
drwxr-xr-x 1 esteban 197121    0 Aug 12 13:53  ./
drwxr-xr-x 1 esteban 197121    0 Jul 30 18:51  ../
-rwxr-xr-x 1 esteban 197121 1.6K Jun 11 20:39 'Warcraft III - Acceso directo.lnk
'*
drwxr-xr-x 1 esteban 197121    0 Aug 12 13:53  compu2-tesis/
-rw-r--r-- 1 esteban 197121  282 Jul 30 18:52  desktop.ini
-rwxr-xr-x 1 esteban 197121 1.3K May 26 17:14 'launcher - Acceso directo.lnk'*

esteban@DESKTOP-ULU1DJS MINGW64 ~/Desktop
$
```

Figura 3.28 Creación de directorio

En el repositorio que se encuentra en la parte de anexos se puede encontrar el proyecto que contiene el *dockerfile*,*Docker-compose* y todos los archivos que se utilizó dentro de la aplicación “restaurante-lopez” que se puede apreciar en la Figura 3.29.

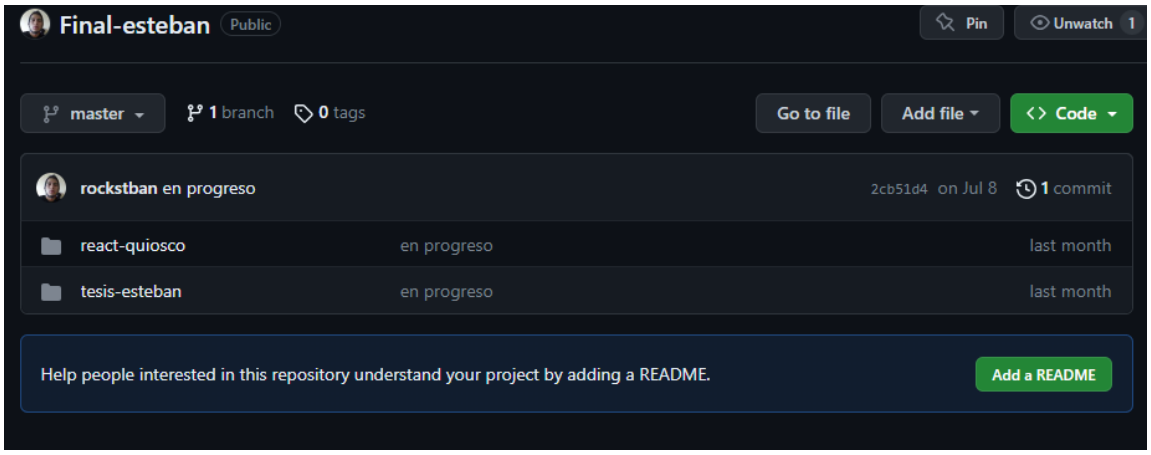


Figura 3.29 Repositorio en *git*

Cuando se sincronice el proyecto con la computadora actual se ingresa a la carpeta establecida en la Figura 3.29 y se verifica que se tenga todos los archivos con el comando “**ls -alh**” como se puede apreciar en la Figura 3.30.

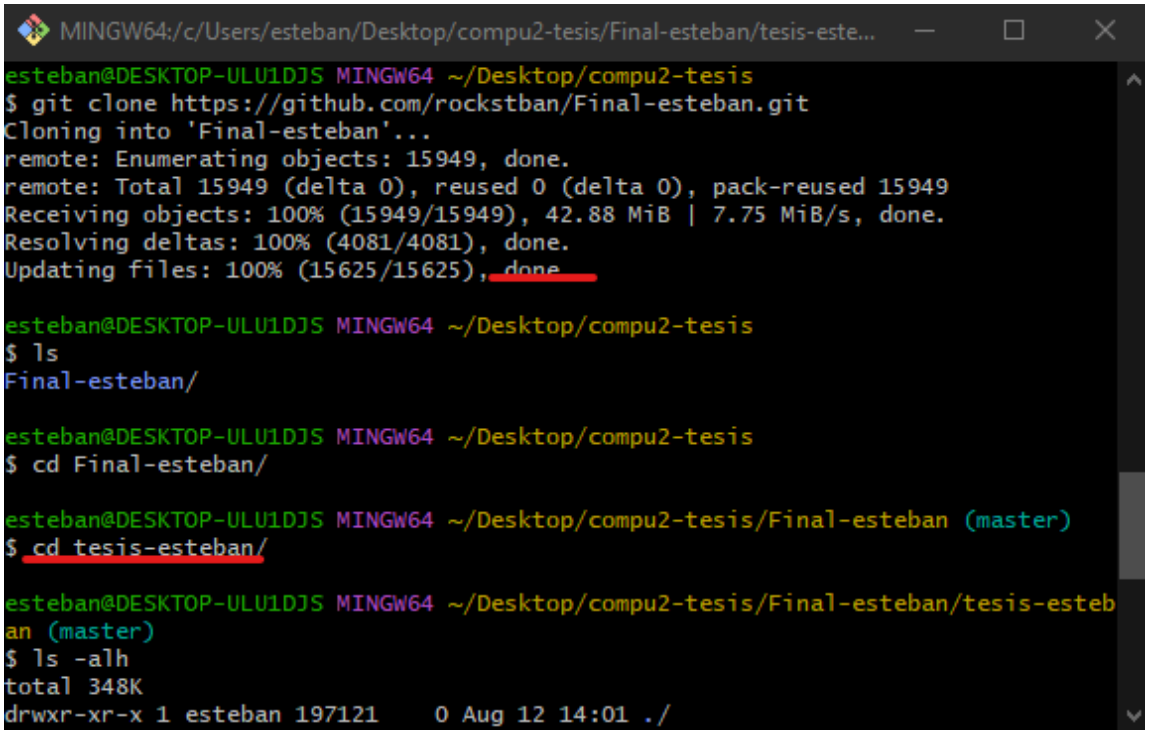


Figura 3.30 Prueba de funcionamiento en otro computador

Se verifica todos los archivos que estén correctamente copiados, cabe recalcar que el archivo señalado en rojo en la Figura 3.31, debe modificarse de acuerdo con las preferencias del usuario final para que el proyecto se levante en la computadora.

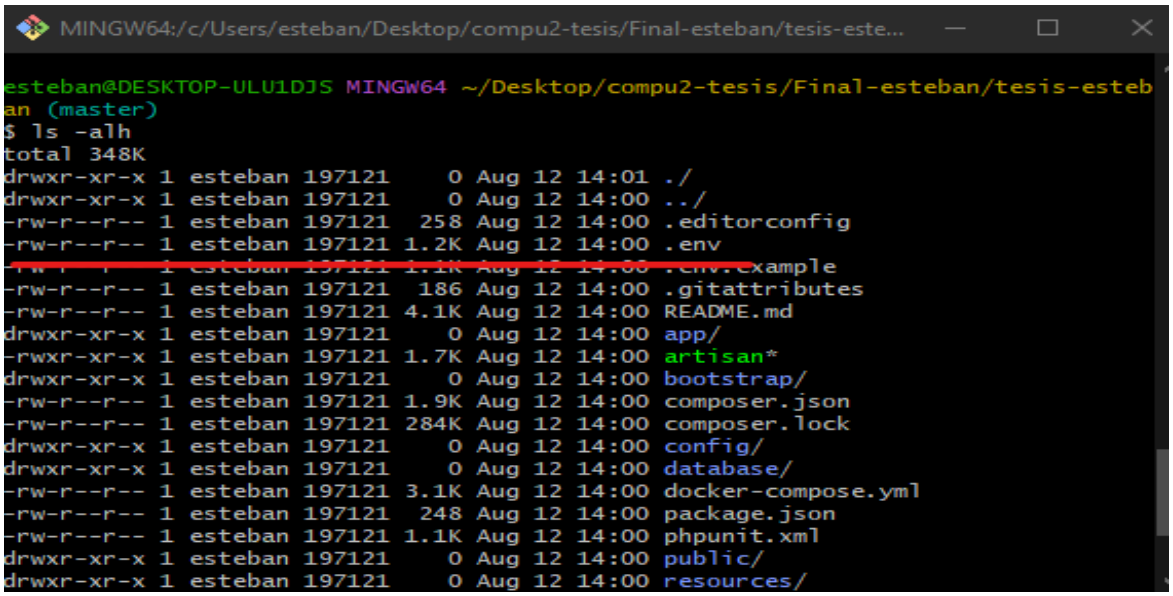


Figura 3.31 Verificación de todos los archivos copiados correctamente

Así como se muestra en la Figura 3.32 se debe colocar su propio usuario y contraseña para la conexión con la base de datos, esto no se encuentra colocado en el proyecto ya que se creó un *gitignore* dado que sería un fallo de seguridad enviar toda la información.

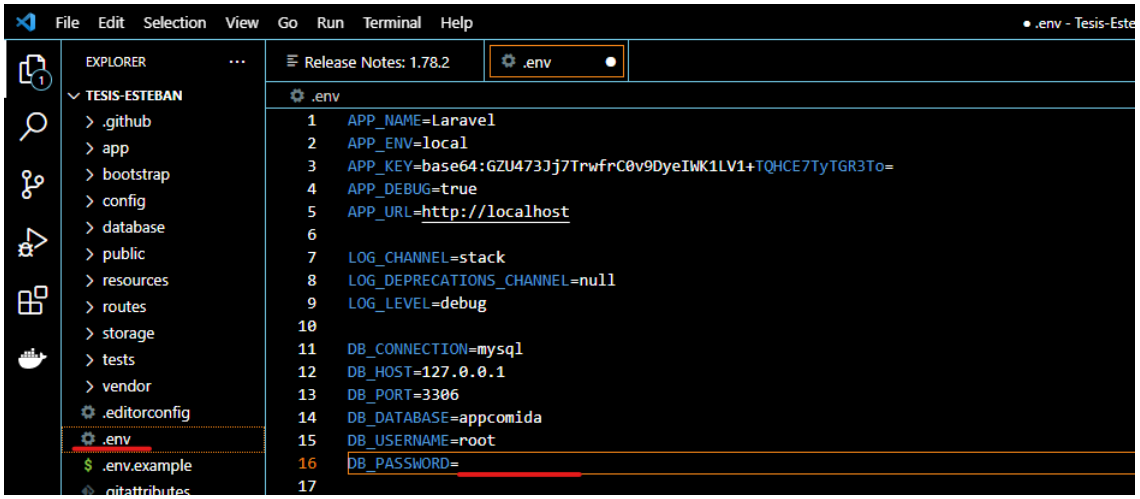


Figura 3.32 Archivo env segunda computadora

Finalmente se levanta el proyecto con el comando mostrado en la Figura 3.33 y se verifica que la aplicación funciona.

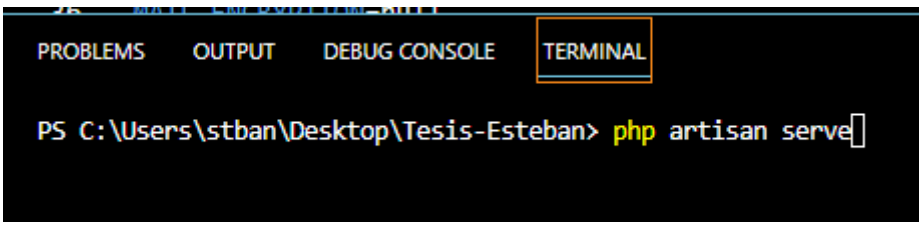


Figura 3.33 Levantar el proyecto computadora dos

4 CONCLUSIONES

- La implementación de herramientas *DevOps* en el proyecto de titulación ha permitido una administración eficiente y confiable de la infraestructura de desarrollo de la aplicación *web*.
- Los entornos destinados al desarrollo y despliegue de servicios y aplicaciones web mediante herramientas *DevOps*, brindan la posibilidad de lograr resultados eficientes, rápidos y seguros en el funcionamiento de servicios.
- La implementación de nuevas herramientas en el proyecto genera que el Dockerfile tenga que volver a ser escrito para que la aplicación pueda ser encapsulada sin generar problemas al momento del despliegue.
- Al mantener un entorno encapsulado se reduce la posibilidad de conflictos entre diferentes versiones de Laravel, permitiendo el despliegue de la aplicación en cualquier ambiente de desarrollo.
- *Docker* es una herramienta que se destaca en entornos que requieren un uso más eficiente de recursos en comparación con las máquinas virtuales. Mientras que las máquinas virtuales ofrecen entornos más robustos, pero a menudo más lentos, con tiempos de despliegue más prolongados, los contenedores proporcionan una mayor facilidad para crear y desplegar entornos de desarrollo completos de manera más rápida y eficiente.
- *Docker* trabaja en conjunto con un *Daemon* el cual genera huecos de seguridad ya que al iniciar un contenedor lo realiza como usuario *root*, generando que cualquier usuario normal pueda ingresar al grupo de *Docker* y realizar en cambios dentro del sistema.
- *Docker* permite tener una compatibilidad con cualquier sistema operativo en donde se encuentre desplegado sin tener problemas de versión con las aplicaciones ya que el contendor se despliega de forma independiente.
- La herramienta *composer* utilizada en el *Framework Laravel* ha permitido la gestión de dependencias para optimizar el flujo de trabajo de la aplicación.
- La implementación de *Docker* ha simplificado notablemente el desarrollo como el despliegue de servicios de aplicaciones web, reduciendo la carga de los recursos necesarios en el sistema anfitrión.

- El despliegue de un contenedor dentro de este trabajo de titulación se realiza mediante la creación de un archivo de configuración Dockerfile, siendo este el que permite crear la aplicación web el cual es ejecutado a través de la herramienta Docker, demostrando que las herramientas DevOps permite mejorar los procesos de implementación dentro de las áreas de TI.

5 RECOMENDACIONES

- Es recomendable optimizar las imágenes *Docker* que son utilizadas en el proceso de despliegue ya que esto permite reducir el tamaño de la imagen eliminando archivos que no sean considerados fundamentales.
- Implementar herramientas de monitoreo y automatización como pipelines permite supervisar el rendimiento de la aplicación desplegada dado que se tiene un control y funcionamiento óptimo en todo momento.
- Realizar investigaciones sobre las formas actuales que se tiene para encapsular una aplicación, ya que si bien *Docker* es una excelente opción existe más herramientas que son utilizadas en ambientes laborales para el despliegue de aplicaciones.
- Realizar un script con todos los comandos de *Docker* es una buena opción para que los lectores aprendan y comprendan como es la construcción de una imagen mediante línea de comandos.
- Es importante recordar que antes de instalar cualquier herramienta dentro del Dockerfile es importante actualizar el repositorio mediante **sudo apt upgrade** y un **update.**
- Finalmente, es importante recalcar la importancia del trabajo en equipo ya que parte de ser profesional corresponde el saber comprender cuando se ha cometido un fallo y se necesita la ayuda de terceros para sacar un proyecto adelante.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] Azure, «¿Qué es DevOps?,» Microsoft, 12 Julio 2021. [En línea]. Available: https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-devops. [Último acceso: 13 Junio 2023].

[2] RedHat, «What is Docker?,» Redhat, 20 Enero 2023. [En línea]. Available: https://www.redhat.com/es/topics/containers/what-is-docker. [Último acceso: 14 Julio 2023].

[3] Microsoft, «Conatiners,» Microsoft, 05 Mayo 2023. [En línea]. Available: https://learn.microsoft.com/en-us/azure/container-apps/containers. [Último acceso: 20 Junio 2023].

[4] Yair, «Qué es composer y como utilizarlo,» Styde, 23 Diciembre 2019. [En línea]. Available: https://styde.net/que-es-composer-y-como-usarlo/. [Último acceso: 15 Julio 2023].

[5] Desarrolloweb, «Laravel,» desarrollodevs, 23 Julio 2020. [En línea]. Available: https://desarrolloweb.com/home/laravel. [Último acceso: 12 Julio 2023].

[6] Microsoft, «Use Docker Compose to deploy multiple containers,» Microsoft, 07 Julio 2023. [En línea]. Available: https://learn.microsoft.com/en-us/azure/ai-services/containers/docker-compose-recipe. [Último acceso: 20 Junio 2023].

[7] Keepcoding, «¿Qué es DockerFile?,» Keep, 18 mayo 2023. [En línea]. Available: https://keepcoding.io/blog/que-es-dockerfile/. [Último acceso: 10 Junio 2023].

[8] J. garzas, «Entendiendo docker. Conceptos basicos,» 31 Julio 2015. [En línea]. Available: https://www.javiergarzas.com/2015/07/entendiendo-docker.html. [Último acceso: 14 Julio 2023].

[9] Redhat, «PODMAN,» Redhat, 21 Julio 2022. [En línea]. Available: https://www.redhat.com/es/topics/containers/what-is-podman . [Último acceso: 10 Agosto 2023].

[10] Arimetrics, «Qué es Framework,» Arimetrics , 14 Febrero 2016. [En línea]. Available: https://www.arimetrics.com/glosario-digital/framework. [Último acceso: 12 Junio 2023].

[11] «Que es Symfony,» Qualitydevs , 05 Agosto 2019. [En línea]. Available: https://www.qualitydevs.com/2019/08/05/que-es-symfony/. [Último acceso: 23 Julio 2023].

[12] CakePHP, «Que es Cake PHP y como utilizarlo,» Cakephp, 03 Enero 2023. [En línea]. Available: https://book.cakephp.org/1.3/es/The-Manual/Beginning-With-CakePHP/What-is-CakePHP-Why-Use-it.html. [Último acceso: 20 Junio 2023].

[13] CoriaWeb, «Que es mvc,» Coriaweb, 14 abril 2018. [En línea]. Available: https://www.coriaweb.hosting/codeigniter-cuales-algunas-ventajas/. [Último acceso: 20 Junio 2023].

[14] Keepcoding, «¿Qué es una imagen Docker?,» keep, 06 Enero 2023. [En línea]. Available: https://keepcoding.io/blog/que-es-una-imagen-en-docker/. [Último acceso: 01 Agosto 2023].

[15] NetAPP, «Que son contenedores,» Netapp, 12 Noviembre 2022. [En línea]. Available: https://www.netapp.com/es/devops-solutions/what-are-containers/. [Último acceso: 12 Agosto 2023].

[16] D. Ochobits, «Colaborativo,» Coloborativo , 09 Febrero 2018. [En línea]. Available: https://colaboratorio.net/davidochobits/sysadmin/2018/crear-imagenes-medida-docker-dockerfile/. [Último acceso: 10 Agosto 2023].

7 ANEXOS

ANEXO I. Certificado de originalidad

ANEXO II. Enlaces

ANEXO III. Conjunto de datos extensos

ANEXO I: CERTIFICADO DE ORIGINALIDAD

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 28 de Agosto de 2022

De mi consideración:

Yo, Fernando Vinicio Becerra Camacho, en calidad de Director del Trabajo de Integración Curricular titulado CREACIÓN DE UNA APLICACIÓN CON DOCKER MEDIANTE UN FRAMEWORK DE PHP elaborado por el estudiante ESTEBAN GABRIEL LOPEZ BASANTES de la carrera en TECNÓLOGÍA SUPERIOR EN REDES Y TELECOMUNICACIONES, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito completo, producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 12%.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: Se adjunta el link del informe generado por la herramienta Turnitin.

https://epnecuador-my.sharepoint.com/:b:/g/personal/fernando_becerrac_epn_edu_ec/EcE5A3c8TVxLrndaa_PZEXABPl_o7JZmmcHKSLK34UHKYA?e=TwB3de

Atentamente,

Fernando Vinicio Becerra Camacho

Docente

Escuela de Formación de Tecnólogos

ANEXO II: ENLACES

Video del proyecto: 5.-Video Funcionamiento.mp4

Repositorio del proyecto: https://github.com/rockstban/Final-esteban.git

Anexo II.I Código QR de la implementación y pruebas de funcionamiento

ANEXO III: CÓDIGOS FUENTE

```
#SE USA LA IMAGEN BASE DE PHP CON APACHE

FROM PHP:8.1-APACHE

#SE COLOCA UNA ETIQUETA PARA IDENTIFICAR QUIEN CREÓ LA IMAGEN

LABEL MAINTAINER="ESTEBAN LÓPEZ"

# SIEMPRE SE ACTUALIZA EL SISTEMA E INSTALA DEPENDENCIAS

RUN APT-GET UPDATE && APT-GET INSTALL -Y \

   LIBPNG-DEV \

   LIBJPEG-DEV \

   ZIP \

   UNZIP \

   GIT \

   && DOCKER-PHP-EXT-CONFIGURE GD --WITH-JPEG \

   && DOCKER-PHP-EXT-INSTALL GD PDO PDO_MYSQL


# SE CONFIGURA EL DIRECTORIO DE TRABAJO DEL PROYECTO

WORKDIR /VAR/WWW/HTML


# COPIA LOS ARCHIVOS DEL PROYECTO

COPY . .


# SE INSTALA LAS DEPENDENCIAS DE COMPOSER

RUN CURL -SS HTTPS://GETCOMPOSER.ORG/INSTALLER | PHP-------INSTALL-
DIR=/USR/LOCAL/BIN --FILENAME=COMPOSER

# SE INSTALA COMPOSER

RUN COMPOSER INSTALL

# COPIA EL ARCHIVO DE CONFIGURACIÓN DEL SERVIDOR APACHE
```

```
COPY APACHE-CONFIG.CONF /ETC/APACHE2/SITES-AVAILABLE/000-DEFAULT.CONF

# HABILITA EL MÓDULO DE REESCRITURA DE APACHE

RUN A2ENMOD REWRITE

# SE EXPONE LA APLICACIÓN EN EL PUERTO 8000

EXPOSE 8000

# AQUI SE PUEDE INICIAR EL SERVIDOR APACHE CON OTRAS VARIABES

CMD ["APACHE2-FOREGROUND"]
```